Master 1 IFPRU
Université de Montpellier II
2007-2008

TER Un logiciel pour rechercher les cycles aromatiques d'une molécule

Anthony DELVECCHIO, Romain POCHARD, Mathieu RÂÏ, Fabien TOURVIEILLE

25 avril 2008

Table des matières

1 Introduction **4**

2 Analyse du sujet **5**

2.1 Présentation du sujet 5

2.2 Quelques notions de chimie et de chimio-informatique 5

2.3 Etude de l'existant 6

2.4 Cahier des charges 7

3 Organisation du travail **8**

4 Développement **10**

4.1 Choix d'une librairie de dessin moléculaire 10

4.2 La librairie ReSyn 11

4.3 Les plugins MarvinSketch 12

4.4 Intégration des algorithmes 12

4.5 Modélisation de l'application 14

4.6 Développement de l'interface 15

4.7 La liaison MarvinSketch-ReSyn 16

4.8 Les différents problèmes techniques rencontrés 17

5 Résultats obtenus **18**

6 Conclusion et perspectives **21**

7 Annexes **22**

7.1 Approfondissement des définitions chimiques et chimio-informatiques 22

7.2 Mode d'emploi de création d'un plugin 23

Table des figures

2.1 Molécule de benzène 6

4.1 Diagramme de classes 14
4.2 Schéma de l'interface de l'application 15

5.1 Capture d'écran du logiciel MarvinSketch 18
5.2 Ecran de paramètres 19
5.3 Visualisation d'un cycle anti-aromatique 19
5.4 Visualisation du premier des deux cycles aromatiques de type R2 20
5.5 Visualisation du second des deux cycles aromatiques de type R2 20

7.1 Strucutres de Kékulé possibles sur un benzène 22

Remerciements

Nous remercions l'Université Montpellier II de nous permettre de mettre en pratique toutes les connaissances acquises lors des différents enseignements dispensés au sein de l'Université de Montpellier II au travers de projets enrichissants tel que ce *plugin* pour MarvinSketch.

Nous tenons à remercier notre tuteur Philippe VISMARA pour sa disponibilité et les conseils qu'il a su nous donner afin d'orienter le groupe, notamment sur le fonctionnement de ReSyn ainsi que les explications sur les notions de chimie.

1 Introduction

Au cours du second semestre de Master 1 Informatique, nous devions réaliser un des projets proposés par des enseignants-chercheurs de l'Université Montpellier II. Le projet intitulé "Logiciel de recherche de cycles aromatiques" a particulièrement retenu notre attention car nous pensions qu'il se rapproche de ce que nous rencontrerons dans le milieu professionnel. En effet, il nécessite des notions dans des domaines nous étant inconnus mais aussi la compréhension d'une application déjà existante.

De plus, ce projet répondait à un réel besoin de chercheurs en chimie, puisque aucune des solutions existantes n'étaient satisfaisantes aussi bien du point de vue de l'algorithme de recherche que de la visualisation des résultats. Pour répondre à ce problème, le LIRMM à développer un algorithme performant que notre application utilisera et aura pour but de diffuser.

La gestion d'un tel projet doit nous permettre de mettre en œuvre les différentes techniques de génie logiciel apprises pendant notre cursus ainsi que les différentes connaissances sur les graphes, l'interface homme-machine ou encore la programmation en JAVA. Les objectifs de ce projet étaient de trouver une librairie permettant de dessiner des graphes moléculaires ainsi que de visualiser des sous graphes partiels correspondants aux motifs, comme les cycles aromatiques, perçus par l'algorithme.

Tout au long de notre rapport, nous répondrons à cette question en étudiant la phase d'analyse, puis celle de développement et enfin les perspectives d'évolution.

2 Analyse du sujet

2.1 Présentation du sujet

Notre sujet portait sur la conception d'un logiciel de visualisation des cycles aromatiques d'une molécule. Nous avions, pour cela, à notre disposition un algorithme basé sur la recherche de cycles (aromatiques, anti-aromatiques) comportant une suite alternée d'arêtes simples ou doubles.
Par la suite, notre travail pourrait être étendu à d'autres algorithmes fournissant des résultats similaires (cycles pertinents). Nous devons utiliser une librairie d'affichage des cycles moléculaires ainsi qu'une bibliothèque de traitement de ces graphes écrite en JAVA.

Notre but est d'arriver à concevoir une application dont l'affichage des résultats obtenus est pertinent. De plus, cette application doit être simple d'utilisation avec une prise en main rapide par les chimistes, leur évitant ainsi une perte de temps.

2.2 Quelques notions de chimie et de chimio-informatique

Tout d'abord pour bien comprendre notre sujet, il faut savoir qu'en chimie organique on représente souvent les molécules par des graphes doublement étiquetés, les sommets correspondants aux atomes (carbone, oxygène, ...) et les arêtes décrivant les liaisons (simple, double, ...). Étant donné un tel graphe moléculaire, on désire déterminer l'ensemble des cycles (aromatiques, pertinents, autres ...) qu'il contient.

Nous allons maintenant aborder diverses définitions de chimie organique telles qu'une molécule, les liaisons, un cycle aromatique, ainsi qu'un cycle pertinent.

MOLECULE : Une molécule est un assemblage d'atomes relié par des liaisons et définie par son graphe moléculaire.

LIAISON : La liaison chimique est le phénomène physique qui lie les atomes entre eux en échangeant ou partageant un ou plusieurs électrons ou par des forces électrostatiques. Ces liaisons peuvent être simples, doubles ou triples c'est-à-dire que le nombre d'électrons participants est de deux, quatre ou six.

CYCLES AROMATIQUES ET ANTI-AROMATIQUES : Un cycle aromatique peut être identifié par une sous structure composée d'une alternance de liaisons simples et doubles, généralement entre des atomes de carbone, ayant un nombre d'arête égal à $4n+2$ avec $n > 0$. Si le cycle comporte $4n$ arêtes, il est dit anti-aromatique. Il peut exister plusieurs cycles aromatiques ou anti-aromatiques de tailles égales ou différentes pour une même molécule. Le plus simple de ces composés, qui ne comporte qu'un seul cycle aromatique de longueur 6 est le benzène.

FIG. 2.1 – Molécule de benzène

CYCLES PERTINENTS : Un cycle est pertinent s'il n'est pas la somme (union disjointe) de cycles strictement plus petits. Si l'on cherche tous les cycles élémentaires d'une molécule, il peut y en avoir un très grand nombre. C'est pour cela que l'algorithme des cycles pertinents a été implémenté.

2.3 Etude de l'existant

Notre tuteur, Philippe Vismara, a eu l'occasion de travailler en collaboration avec des chimistes et connaît donc les différentes applications existantes. Il existe des logiciels propriétaires et payants telle qu'ISISDraw qui est largement diffusée. Dans le domaine du logiciel libre on peut citer la suite des logiciels Marvin de la société Chemaxon sortant peu à peu de l'anonymat et la librairie OpenSource Java « The Chemistry Development Kit » utilisée par l'application JChem.

Cependant, elles ne sont pas satisfaisantes en termes de calcul d'aromaticité car elles couvrent peu de cas. Leur algorithme de recherche de cycles est peu efficace et incomplet. Elles ne comportent pas non plus de visualisation efficace des cycles obtenus. C'est pourquoi le LIRMM (Laboratoire d'Informatique, de Robotique et Microélectronique de Montpellier), en association avec des chimistes, a développé sa propre application ReSyn, comportant entre autres un nouvel algorithme de recherche

de cycles aromatiques ainsi qu'une interface graphique de représentation des molécules.
ReSyn est une application qui demeure peu conviviale, difficile à prendre en main et compliquée d'utilisation du fait de son trop grand nombre de fonctionnalités.

2.4 Cahier des charges

Comme nous l'avons vu dans le paragraphe précédent, il existe un certain nombre de solutions pour la visualisation des cycles aromatiques d'une molécule mais aucune d'elle n'est exempte de défauts.

Par exemple :

Marvin n'offre qu'une gestion de la recherche de cycles aromatiques très sommaire, un algorithme peu efficace et une visualisation peu lisible. ReSyn est difficile à utiliser du fait d'une interface très lourde en options diverses et peu intuitive, ce qui oblige les chimistes à un long apprentissage. JChem, quant à lui, implémente un algorithme de recherches de cycles mais qui est peu efficace et contient une interface peu conviviale.

L'objectif principal est la réalisation d'une application capable de trouver et de représenter les cycles aromatiques de certaines molécules chimiques. Les objectifs secondaires sont :
- l'intégration des algorithmes permettant la visualisation d'autres structures moléculaires (cycle pertinents).
- une utilisation des algorithmes de recherches de cycles en ligne de commande.
- la disponibilité de l'application directement sur le web sous forme d'applet
- et éventuellement l'implémentation d'un algorithme de générations de structures de Kékulé d'une molécule.

Le but de ce TER sera donc d'analyser les avantages des solutions existantes pour concevoir une application permettant un affichage pertinent des résultats. Pour cela nous disposons des algorithmes efficaces de recherches de cycles de ReSyn ainsi que la possibilité de rencontrer des chimistes et de s'adapter à leurs attentes.

3 Organisation du travail

Contrairement aux projets précédemment réalisés, l'équipe de travail de celui-ci se compose de quatre membres. Il a donc été primordial de bien se répartir les tâches pour que le projet se déroule le plus efficacement possible. Outre les différents rôles que nous nous sommes attribués au sein du projet tel que chef de projet ou encore responsable communication, nous nous sommes scindés en deux sous groupes de travail.
Le premier groupe travaillait sur l'interface homme-machine et l'ergonomie du logiciel tandis que le second s'est chargé de l'intégration des différents algorithmes de ReSyn dans la librairie graphique choisie.

Cette répartition en binôme nous a permis de nous initier au Pair Programming qui est une des méthodes de l'eXtrem Programing (une des méthodes agiles). Dans cette méthode, le premier programmeur, appelé pilote, a le clavier et travaille sur la portion de code à écrire. Le second, appelé copilote, est là pour l'aider, en suggérant de nouvelles possibilités ou en décelant d'éventuels problèmes.
Les rôles s'échangent régulièrement pendant la séance de programmation. Ces interactions permettent une bonne connaissance de l'application et une bonne communication au sein de l'équipe.
De plus cette répartition nous a donné la possibilité de se voir régulièrement sans rencontrer de problème majeur de disponibilité. Elle a également amené une certaine objectivité quant au travail réalisé par l'autre sous-groupe, permettant ainsi d'apporter notre point de vue, nos remarques.

Nous avons également eu l'occasion de rencontrer les personnes susceptibles d'utiliser le logiciel. Durant ces réunions, nous avons appris quelles étaient les principales attentes des chimistes, ce qui nous a permis de nous concentrer sur les points les plus importants à développer. Les réunions avec notre tuteur étaient au départ prévues tous les 15 jours mais se sont faites plus présentes sur la fin allant jusqu'à deux fois par semaine.

Nous avions pensé créer un serveur SVN (gestionnaire de dépôt de versions centralisé) que nous avions pu utiliser dans d'autres projets, mais l'idée a vite été abandonnée au vue de notre distribution

des tâches qui nous permettait de travailler sans risque de corruption du travail de l'autre groupe.

4 Développement

4.1 Choix d'une librairie de dessin moléculaire

Après avoir étudié l'existant et les différentes solutions offertes aux chimistes, nous nous sommes rendu compte qu'il était fastidieux de créer notre propre application. En effet, les applications existantes permettent de dessiner une molécule, de la visualiser en 3D ainsi que de nombreuses autres fonctionnalités ; ce que nous n'aurions pas pu égaler dans le temps imparti.

L'idée principale étant de diffuser l'algorithme de recherche de cycles aromatiques développé par le LIRMM, l'emploi d'une librairie ou d'une application « mère » pouvant recevoir notre application semblait la solution la plus intéressante.

Le calcul des cycles étant implémenté dans les classes de la librairie ReSyn développée en JAVA, nous avons donc étudié les différents outils développés dans le même langage. Notre choix s'est concentré sur la création d'un *plugin* pour MarvinBeans de la société Chemaxon ou bien l'utilisation de la librairie CDK (Chemistry Development Kit).

Nous avons opté pour le développement d'un *plugin* pour le logiciel MarvinSketch pour plusieurs raisons :
- MarvinSketch dispose d'un *framework* pour l'ajout de *plugins* permettant d'effectuer des calculs sur les molécules éditées par le logiciel.
- MarvinSketch est un logiciel de chimie comprenant une interface graphique pour le dessin de molécules, de plus elle est nettement plus développée que CDK. Elle permet aussi de manipuler des fichiers au format MDL (Molecular Design Limited) fréquemment utilisé par ce type de logiciel.
- MarvinSketch fait partie des logiciels de plus en plus utilisés dans le domaine de la chimio-informatique, notre projet pourra donc être utilisé par une communauté d'utilisateurs conséquente alors que CDK est plus confidentiel.

4.2 La librairie ReSyn

La partie de la librairie ReSyn que nous utilisons se décompose en deux packages, celui correspondant aux structures de graphes moléculaires et celui correspondant aux calculs des différents algorithmes.

Voici les principales classes de structures des graphes moléculaires :

UGraph : C'est la classe mère des graphes moléculaires. C'est une classe de graphe universelle et non spécifique à notre projet. Il suffit de renseigner son constructeur avec le nombre de sommets, et deux booléens indiquant si les sommets et arêtes sont étiquetés.

UMolGraph : Cette classe hérite de UGraph, elle précise que ce graphe est un graphe moléculaire en intégrant des fonctions d'étiquetage spécifiques aux graphes moléculaires ou des informations supplémentaires comme la charge.

Bloc : Un Bloc n'est rien d'autre qu'un sous graphe partiel d'un UGraph.

AtomType : Cette classe s'occupe de la gestion des atomes, elle peut, par exemple, nous donner le symbole chimique d'un atome, elle nous est utile pour étiqueter les labels sur les sommets.

Util : La classe Util est nécessaire pour le chargement de fichiers XML, permettant les différentes initialisations de ReSyn comme les données sur les types d'atomes.

Les classes utilisées pour les calculs des différents algorithmes :

BlocPerception : C'est une classe abstraite définissant les fonctions communes aux algorithmes de perception.

AromaticAlgoNode : C'est la classe permettant de trouver les cycles aromatiques et anti-aromatiques pour une molécule donnée. Il suffit de passer en paramètres du constructeur notre graphe (UMolGraph) ainsi que deux booléens donnant la possibilité de paramétrer l'algorithme de recherche.

CyclesPerti : Cette classe permet de trouver les cycles pertinents pour une molécule donnée. Son constructeur n'a besoin que du graphe moléculaire pour exécuter l'algorithme de calcul.

4.3 Les plugins MarvinSketch

Marvin dispose d'une structure de classes permettant de lui ajouter des *plugins*. Ainsi, on peut adapter l'application aux besoins spécifiques d'un utilisateur en intégrant ses calculs et algorithmes dans le logiciel MarvinSketch. Nous avons rencontré quelques difficultés à la création de ce *plugin* car la documentation existante est très sommaire. De plus, nous n'avions qu'un seul exemple à disposition, qui ne présentait qu'un aspect simpliste des possibilités offertes en termes de développement.

Une fois assimilée, cette structure doit permettre de mettre en place un mécanisme pouvant générer une fenêtre de paramétrage du *plugin* (choix de l'algorithme, couleurs d'affichage), d'effectuer un calcul ainsi que de visualiser les résultats obtenus par l'intermédiaire de deux classes CalculationPlugin et CalculationPluginDisplay et d'un fichier XML.
Celui-ci permet de générer une fenêtre de paramètrage de l'application. (voir section 4.6), tandis que les deux classes contiennent les méthodes devant être implémentées dans les classes spécifiques à notre *plugin* pour récupérer les molécules éditées dans le logiciel ainsi que les paramètres saisis dans la fenêtre puis effectuer le calcul et enfin récupérer les résultats pour l'affichage.

Une fois le *plugin* développé, les fichiers .class de notre application, les fichiers XML de configuration de ReSyn, ainsi que le fichier de paramétrage de l'interface doivent être placés dans un fichier JAR qui sera lié à Marvin lors de son lancement suite à l'ajout d'une ligne normalisée dans un fichier de configuration.

4.4 Intégration des algorithmes

La librairie de Chemaxon possède son propre modèle de graphe moléculaire tout comme celle de ReSyn. On dispose donc de deux librairies autonomes et une des principales questions que l'on s'est poser tout au long du développement est : comment fusionner de manière simple et la plus efficace possible ces deux librairies ?
Comme évoqué précédemment, les algorithmes de recherche de cycles sont présents dans la librairie de ReSyn. Une partie importante du travail a été d'intégrer ces algorithmes dans un *plugin* Chemaxon afin de pouvoir l'exécuter sur les molécules affichées, construites ou encore chargées par l'intermédiaire de MarvinSketch. La première partie du travail fut de récupérer la molécule affichée dans la fenêtre principale de Chemaxon. A partir de cette molécule, on doit extraire les données nécessaires et suffisantes à l'exécution des algorithmes de recherches. Pour cela, nous avons eu diverses idées.

La première fut d'enregistrer la molécule au format MDL puis de l'ouvrir ensuite avec la librairie de ReSyn pour exécuter l'algorithme.
Cette méthode requiert l'utilisation de fichiers temporaires ce qui est peu pratique et non recommandable. Elle fut abandonnée très rapidement, sans passer par une phase de développement. Ensuite, nous avons pensé à modifier la librairie ReSyn pour qu'elle s'adapte et soit capable d'exécuter les algorithmes sur des instances de graphes de Chemaxon. Après concertation avec notre tuteur, nous avons également abandonné cette solution car la librairie ReSyn est beaucoup trop lourde et est adaptée à sa propre représentation de molécules qui est très différente de celle proposée par Chemaxon.

Enfin, nous avons opté pour une méthode consistant à générer une instance de graphe de ReSyn à partir de la molécule éditée dans Marvin. Cette dernière consiste en une « traduction » d'un graphe moléculaire qui représente la molécule sous MarvinSketch en un UMolGraph (graphe moléculaire de la librairie ReSyn) qui inclue les structures et les données nécessaires à sa compréhension par la librairie de ReSyn.

Pour cela, on dispose de nombreux accesseurs en lecture et écriture à la fois dans la librairie de MarvinSketch et dans celle de ReSyn nous permettant de récupérer par exemple le nombre d'arêtes d'une molécule, ses labels d'atomes. Cette traduction a été fastidieuse et délicate à mettre en place (voir section 4.8).Nous avons également du faire évolué la librairie de ReSyn, en accord avec notre tuteur, par l'ajout de méthodes permettant de travailler sur des Bloc. Une fois le graphe (i.e. : la molécule) traduit, on peut alors exécuter tous les algorithmes de recherche de cycles qui sont implémentés dans ReSyn. La réalisation de ce travail nous a pris du temps, en rapport avec les nombreux bugs rencontrés tout au long de l'intégration (voir section 4.8).

4.5 Modélisation de l'application

De l'étude de la librairie de ReSyn ainsi que de l'élaboration d'un *plugin* Chemaxon, nous avons pu établir un diagramme de classe résumant le développement de notre *plugin*.

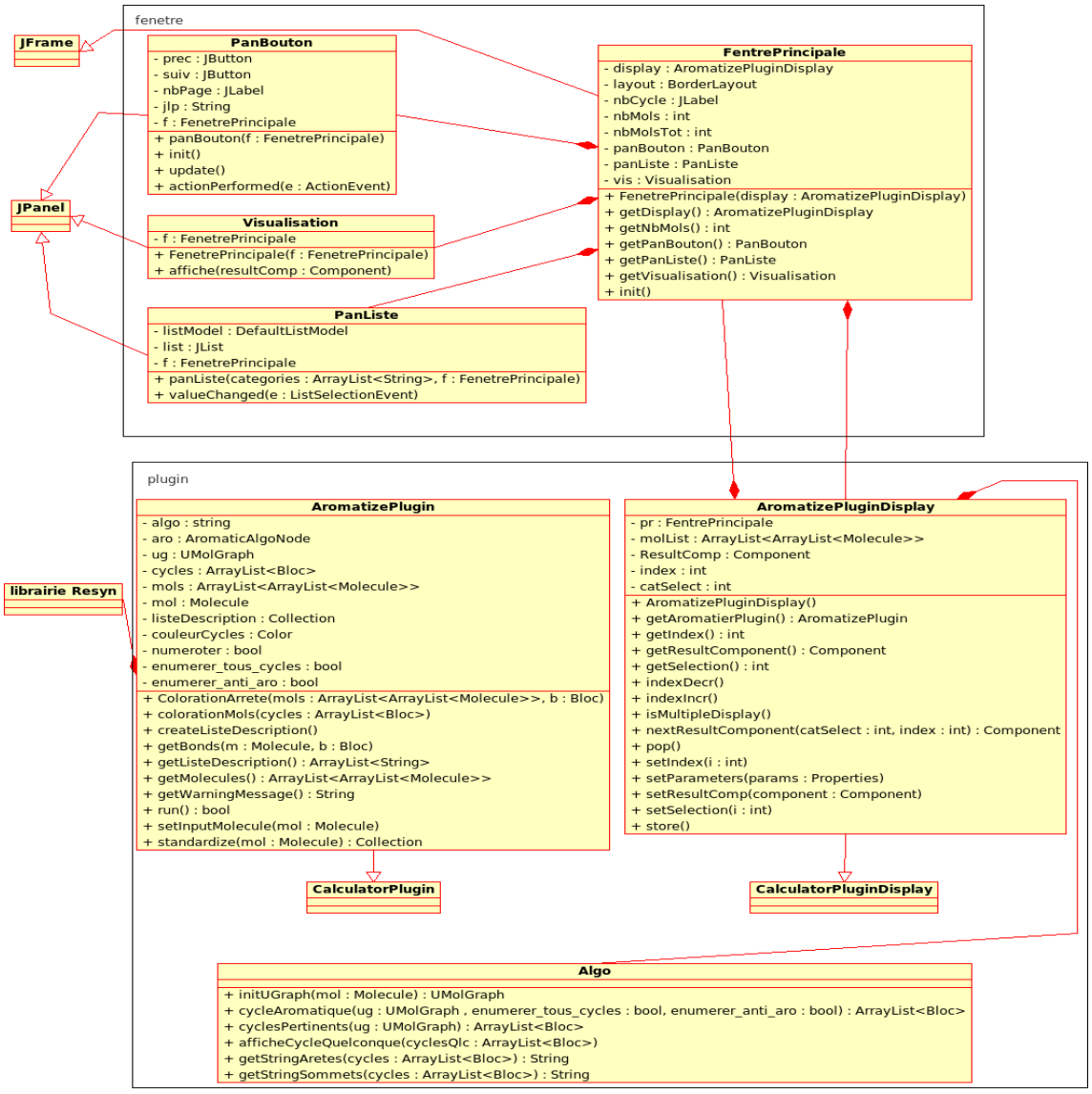

FIG. 4.1 – Diagramme de classes

Le code de l'application est ainsi séparé en différents packages. Les packages « graphe » et « perception » sont issus de la librairie de ReSyn et contiennent les différents algorithmes permettant la recherche de cycles. Le package « plugin » quant à lui contient l'ensemble des classes nécessaires au lancement du *plugin*, à l'interaction entre le logiciel Marvin et la librairie de Resyn ainsi que l'interaction avec le package « fenetre ». Ainsi, ce dernier package contient notre code implémentant l'interface graphique du *plugin* nécessaire à la visualisation des cycles.

4.6 Développement de l'interface

MarvinSketch dispose d'une classe destinée à la représentation d'une molécule sous la forme d'un composant graphique. Dans notre *plugin*, nous avons la possibilité de créer une instance de cette classe pour représenter la molécule éditée dans la fenêtre principale de Marvin. Cependant, nous n'avons aucun pouvoir d'action sur les objets de cette classe, nous ne pouvons donc ni modifier la molécule représentée, ni modifier sa coloration, ni sélectionner une arête. Pourtant, dans notre application, nous devons faire évoluer la molécule de façon à faire ressortir, les unes après les autres, les caractéristiques recherchées. C'est pourquoi nous avons choisi de cloner la molécule à analyser puis de modifier chacun de ses clones indépendamment. Ces modifications portent essentiellement sur la coloration d'arêtes composant le cycle à observer.
Les motifs pertinents calculés par ReSyn disposent d'une description qui les référence dans une certaine catégorie. Dans notre interface, un classement par catégorie est utile aux chimistes lors de l'analyse des résultats obtenus.

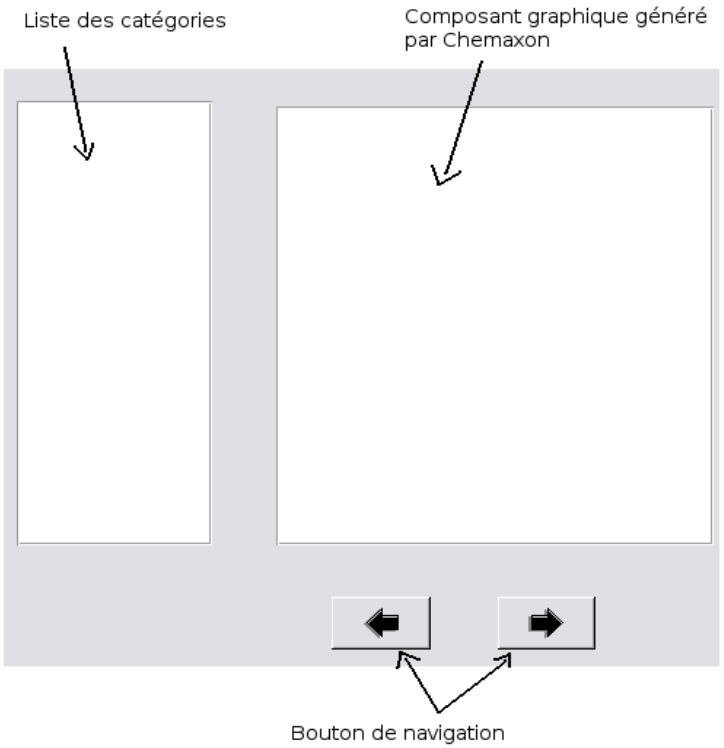

FIG. 4.2 – Schéma de l'interface de l'application

Le package « fenetre » comprend les classes ci-dessous que nous avons développé pour créer l'interface graphique.

La classe Visualisation est un JPanel qui comprend le composant graphique propre à Marvin permettant d'effectuer plusieurs opérations telles que des rotations, zooms, affichage des atomes.

La classe PanListe est un JPanel comprenant une liste, classée par catégorie, des cycles inclus dans la molécule.

La classe PanBouton comprend deux boutons permettant de naviguer dans l'ensemble des molécules.

La classe FenetrePrincipale hérite de JFrame et est ainsi la fenêtre principale de notre *plugin* comportant une instance des 3 classes ci-dessus.

Dans les *plugins* MarvinSketch, une fenêtre de paramétrage peut être générée à partir d'un fichier XML dont le décodage est essentiellement géré par Chemaxon. Ce fichier XML doit respecter un certain formalisme permettant de créer des cases à cocher, boutons radio, listes déroulantes.
Cette fenêtre nous permet de choisir les algorithmes à exécuter ainsi que leurs paramètres (voir section 4.2), mais aussi l'affichage de la numérotation des atomes et enfin la couleur des cycles à mettre en évidence (Notre tuteur ayant des difficultés à percevoir les couleurs, cette option lui était chère).

4.7 La liaison MarvinSketch-ReSyn

La classe Algo comprend les méthodes telles qu'initUGraph(), cyclesAromatique(), cyclesPertinent(). Elles font appel aux méthodes de la librairie ReSyn initialisant les graphes moléculaires et exécutant les algorithmes.

Les classes AromatizePlugin et AromatizePluginDisplay, que nous avons implémentés, héritent respectivement de CalculatorPlugin et CalculatorPluginDisplay qui constituent le cœur du *plugin*. La première peut être vue comme la classe qui effectue le calcul des cycles aromatiques et la seconde comme la classe gérant l'affichage graphique. Dans la classe AromatizePlugin, la méthode setInputMolecule(Molecule mol) permet de récupérer la molécule dessinée par l'utilisateur dans la fenêtre principale de Marvin. La méthode run() effectue la recherche de cycles par l'intermédiaire des méthodes de la classe Algo.

La liaison entre les deux classes citées ci-dessus se fait par l'intermédiaire de la méthode getMolecules() qui retourne un ensemble de molécules. Chaque élément de cet ensemble est un clone de la molécule dessinée à l'écran sur laquelle un des cycles aromatique a été coloré en fonction des cycles calculés par la méthode run() . La classe AromatizePluginDisplay génère la fenêtre graphique de notre application qui est créée et retournée par la méthode getResultComponent(). La méthode nextResultComponent() fait appel à la méthode super.getResultComponent() qui retourne un composant graphique propre à Chemaxon comprenant le dessin de la molécule.

4.8 Les différents problèmes techniques rencontrés

Nous allons donc parler des problèmes techniques rencontrés tout au long du projet. La première difficulté nous est apparue avec la compréhension des termes de chimie. Ainsi, cela n'a pas été évident de comprendre toutes les explications des API Chemaxon et ReSyn mais aussi de la documentation papier fourni par notre tuteur. Il nous a fallu aussi du temps pour comprendre la structure des schémas moléculaires des deux librairies et les faire correspondre. Comme dit précedemment, le manque d'informations sur le développement d'un *plugin* pour Marvin nous a posé des problèmes. Ceux-ci nous ont fait perdre un certain nombre d'heures car nous devions programmer par tâtonnement.

Une fois ceci acquit, la restauration des données par les fichiers XML de paramétrage de ReSyn a été notre plus gros problème. Après maintes recherches, il s'est avéré que le problème venait de la classe XMLDecoder de JAVA. Cette dernière n'arrivait pas à gérer correctement le classpath provoquant ainsi une difficulté avec certaines méthodes de restauration des objets. XMLDecoder ne trouvait donc pas les classes définies dans notre JAR. De ce fait, on ne pouvait pas restaurer les fichiers XML comprenant les objets persistants nécessaires à l'exécution des algorithmes.

Il existe un bug similaire de gestion de classpath dans JAVA, il sera corrigé dans JAVA 1.7. Il a alors fallu demander des modifications, par notre tuteur, de la librairie ReSyn pour résoudre ce problème.

Nous avions pour objectif de réaliser un applet Java de notre *plugin*, comme il en existe sur le site de Chemaxon, mais le peu de documentation à ce sujet sur leur site nous à rendu la tâche impossible.

5 Résultats obtenus

Après vous avoir présenter les différents outils et méthodes de développement, nous allons détailler les résultats obtenus. Afin de bien situer où se trouve notre *plugin*, il est nécessaire de vous présenter la vue générale du logiciel MarvinSketch. À partir de cet interface, on peut : dessiner, charger des molécules, exécuter des calculs sur ces molécules et surtout exécuter les *plugins* à partir du menu Tools.

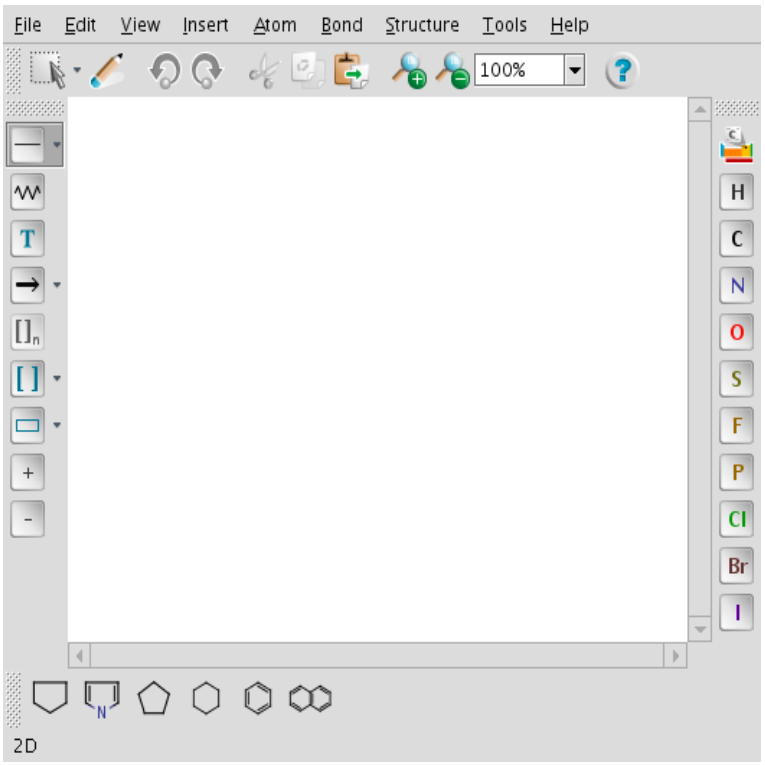

FIG. 5.1 – Capture d'écran du logiciel MarvinSketch

Pour lancer notre *plugin*, il suffit de cliquer sur Tools > Aromatize. L'écran de paramétrage suivant s'affiche.

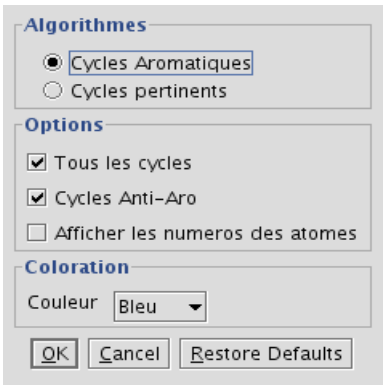

FIG. 5.2 – Ecran de paramètres

Une fois les options sélectionnées, la fenêtre principale de notre *plugin* s'affiche et on peut ainsi visualiser les différents cycles. On voit sur la droite la liste des catégories permettant de classer les cycles selon leurs longueurs. Ici, les arêtes du cycle anti-aromatique de longueur 8 est coloré en bleu.

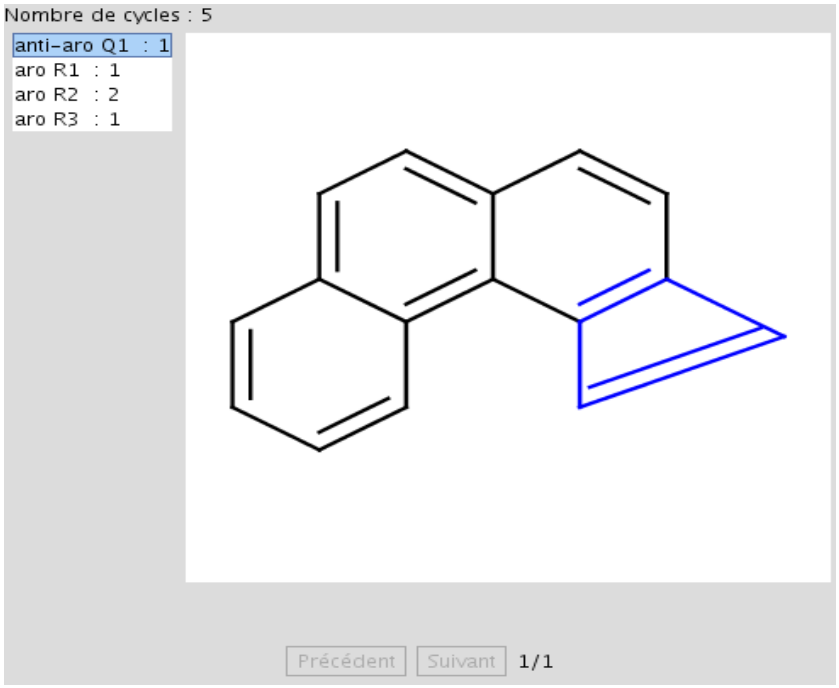

FIG. 5.3 – Visualisation d'un cycle anti-aromatique

En cliquant sur l'item aro R2 dans la liste, on peut visualiser les cycles aromatiques de types R2 de la molécule (au nombre de deux ici).

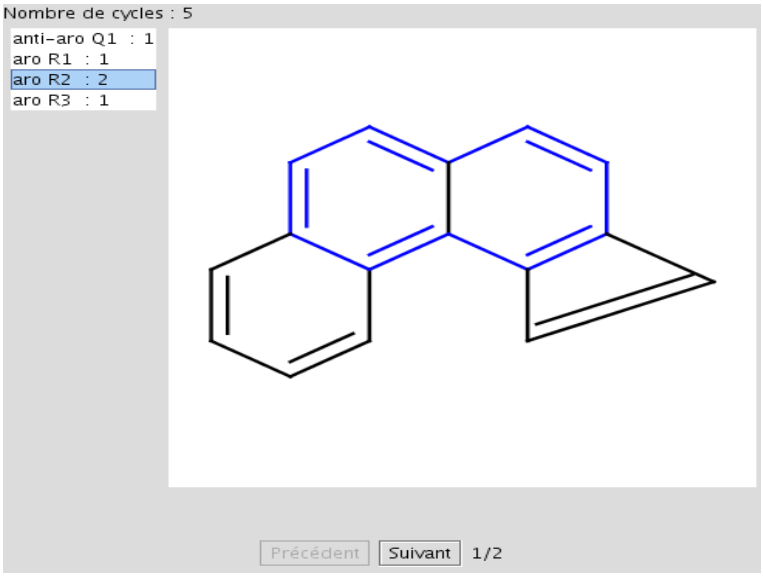

FIG. 5.4 – Visualisation du premier des deux cycles aromatiques de type R2

Les boutons « Précédent » et « Suivant » permettent de visualiser les autres cycles. Ici on voit le deuxième cycle R2 de la molécule.

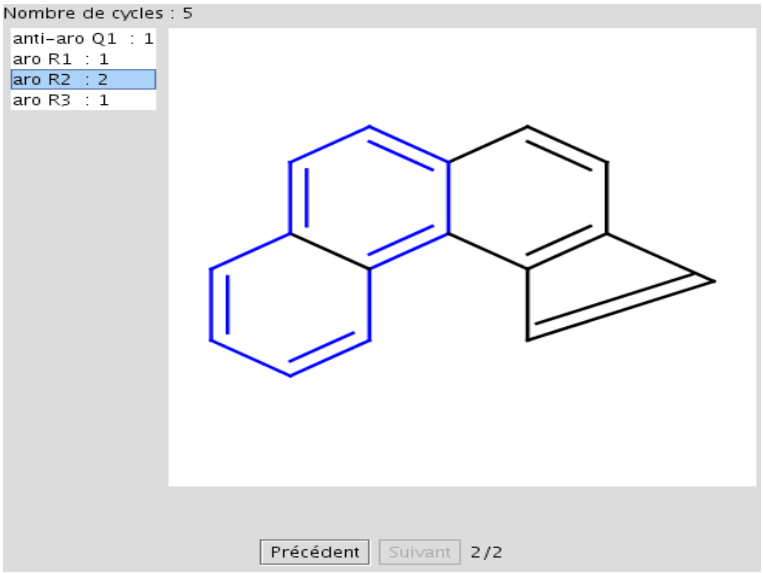

FIG. 5.5 – Visualisation du second des deux cycles aromatiques de type R2

6 Conclusion et perspectives

Les exigences de notre tuteur et des futurs utilisateurs étant grandes, nous avons dû tout au long de ce projet nous efforcer à proposer des solutions leur convenant. Un réel échange développeurs-utilisateurs s'est donc engagé, ce qui à, en grande partie, contribué à rendre ce travail enrichissant.

Notre application propose de visualiser des cycles aromatiques, anti-aromatiques ou pertinents, au sein d'une molécule chimique éditée à partir d'un logiciel de chimie avancé. Nous pensons avoir rempli les objectifs principaux définis par le cahier des charges tels qu'une visualisation efficace et simple ainsi que l'intégration d'un algorithme performant au sein d'un logiciel répandu assurant une partie de la diffusion de ce dit algorithme. Entres autres, cette application a été soumise à l'avis de chimistes, dans l'ensemble satisfaits de la solution proposée.

Le travail réalisé a renforcé nos acquis en matière de programmation JAVA, de gestion de projet et a encore une fois démontré l'importance des graphes en informatique.

Le développement de ce projet nous a également permis de nous projeter dans l'avenir d'une application. En effet, on peut envisager de nombreuses perspectives pour ce *plugin*. La population de chimistes utilisant des logiciels étant somme toute très faible, la traduction du *plugin* en Anglais, en utilisant les mécanismes de support multilingue de JAVA, semble tout à fait appropriée à une diffusion plus large de l'application. MarvinSketch incluant de nombreux *plugins* officiels, il pourrait être intéressant de prendre contact avec ChemAxon pour éventuellement discuter de la possibilité de rendre officiel ce *plugin*. Cette prise de contact permettrait également d'approfondir la question concernant la réalisation de l'applet.

Il pourrait aussi être intéressant de développer l'algorithme énumérant toutes les structures de Kékulé d'une molécule aromatique que nous n'avons pas pu développer par manque de temps mais aussi à cause de sa complexité (article). En effet, à ce jour, aucune application libre ne propose cette énumération.

7 Annexes

7.1 Approfondissement des définitions chimiques et chimio-informatiques

ATOME : Un atome est généralement constitué d'un noyau composé de protons et de neutrons autour desquels se trouvent des électrons. En chimie, les atomes sont les éléments de base. Ils constituent la matière et forment les molécules en partageant des électrons.

CHARGE : La charge électrique d'un atome est définie par le nombre d'électrons et de protons constituant le noyau. Les atomes sont susceptibles de se charger électriquement en gagnant ou en perdant un ou plusieurs électrons : on parle alors d'ions. Du fait qu'un électron a une charge électrique négative, si un atome gagne un ou plusieurs électrons, la charge de l'atome devient négative (anion), et s'il en perd, la charge de l'atome devient positive (cation).

STRUCTURES DE KEKULE : La représentation des cycles aromatiques par des liaisons alternées est en fait inexacte. Il s'agit en réalité d'une circulation d'électrons le long du cycle. (Fig. c) Un même cycle aromatique peut donc être représenté par deux cycles alternés suivant la place des liaisons simple et double. (Fig. a et b) Une molécule contenant un ou plusieurs cycles aromatiques ou anti-aromatiques peut être représentée de plusieurs manières en échangeant les liaisons simples et doubles de ces cycles. Les structures de Kékulé sont ces représentations différentes d'une même molécule. Ces dernières sont importantes pour les chimistes mais difficile à appréhender et trouver.

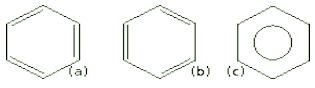

FIG. 7.1 – Strucutres de Kékulé possibles sur un benzène

7.2 Mode d'emploi de création d'un plugin

Pour exécuter le *plugin* sous Marvin, deux opérations doivent être réalisées :
Un JAR doit être créé, il devra comporter les fichiers .class de l'application, les fichier nécéssaires (Pour nous les fichiers XML), et un fichier du nom du *plugin* + Parameters.XML (AromatizePluginParameters.xml dans notre exemple) que l'on place dans un repertoire nommé xjars.
Il faut ensuite rajouter une ligne dans le fichier plugins.properties se trouvant dans le dossier *plugin*, lui même dans le dossier d'installation de Marvin.
Cette ligne est de la forme
$ < *pluginclassname* > $ < *pluginJARURL* > $ < *menu* > $ < *mnemonics* > $ < *group* > $ < *groupmnemonics* > $ < *NOPARAMS* >
Dans notre cas :
*plugin*_9999=$$*AromatizePlugin.jar*$*Aromatize*$*Aromatize*$$

Bibliographie

[1] Ivan GUTMAN et Milan RANDIĆ. *A correlation between Kekulé valence structures and conjugated circuits*. 1978.

[2] Christoph STEINBECK, Yongquan HAN, Stefan kUHN, Oliver HORLACHER, Edgar LUTTMANN et Egon Willighagen. *The Chemistry Development Kit (CDK) : An Open-Source Java Library for Chemo and Bioinformatics*. 2002.

[3] Wikipedia : *http ://fr.wikipedia.org/wiki/Methode_ agile, http ://fr.wikipedia.org/wiki/Extreme_ programming, http ://fr.wikipedia.org/wiki/Programmation_ en_ binôme*.

[4] Le site de ChemAxon : *http ://www.chemaxon.com*.